Improving the Cosmos molecular dynamics program using multi-timestepping techniques

Galen Reeves

Introduction

Multiple-timestepping techniques have the ability to improve the speed of a molecular dynamics simulations. These techniques sample the expensive and slowly changing long-range forces less frequently than the short-range forces. Tuckerman and Berne have developed an integrator called reversible reference system referencer algorithm (rRESPA) [3] based on the Trotter expansion of the Liouville Propagator. The (rRESPA) algorithm separates the long and short range forces in a way that is reversible, and therefore conserves energy and has some degree of stability.

In this report, the Trotter expansion used in (rRESPA)is has been implemented into the Cosmos molecular dynamics program [2] using the forces from the particle mesh Ewald (PME) as the long range forces. The method created, Cosmos-Trotter, is tested on a water-hexane system and a protein system.

Improved speed in molecular dynamics simulations will allow for the study of systems for longer time intervals. Many biological processes occur on the order of milliseconds and the applications of the md program increase with its simulation time.

Implementation

The Trotter expansion fits in well with the basic structure of Cosmos and is able to use the velocity Verlet routine to iterate the system. To run cosmos-Trotter, a short time step, δt, and a long time step, Δt, need to specified. The value of δt is given by the original time step supplied in regular Cosmos, and Δt is given by the parameter *ntrott*, where $ntrott = \Delta t / \delta t$.

The forces are split into short-range and long-range components. The short-range forces are determined from the straightforward calculations of all the atoms up to a given cutoff length. The long-range forces are calculated using the particle mesh Ewald (PME) sum.

Currently, there are several restrictions on the input parameters. On the steps where the PME is not run, the total energy of the system cannot be calculated. Therefore, energy statistics should be taken only on Δt steps. Also, to assure that a run finishes correctly, the total number of steps in the run be a multiple of *ntrott*.

The Algorithm. The program uses the velocity Verlet algorithm to propigate each step. A regular velocity Verlet step on a system takes $x(t) \rightarrow x(t+\delta t)$ and $v(t) \rightarrow$

$v(t+\delta t)$.

$v \leftarrow v + F\frac{\delta t}{2m}$
$x \leftarrow x + v\,\delta t$
calculate F
$v \leftarrow v + F\frac{\delta t}{2m}$

In the Cosmos implementation of velocity Verlet, the order of the algorithm is changed slightly to put the force calculations at the beginning of the step. The explicit implementation consists of the following:

calculate F
$v(t) \leftarrow v(t - \frac{\delta t}{2}) + F(t)\frac{\delta t}{2m}$
perform velocity rattle
$v(t + \frac{\delta t}{2}) \leftarrow v(t) + F(t)\frac{\delta t}{2m}$
$x(t+\delta t) \leftarrow x(t) + v(t + \frac{\delta t}{2})\frac{\delta t}{2m}$
perform position and velocity shake

Throughout a single step,$F(t)$ has the same value. A step still takes $x(t) \rightarrow x(t+\delta t)$. However, the velocities of the system are taken from $v(t - \frac{\delta t}{2}) \rightarrow v(t + \frac{\delta t}{2})$. When the total energy for a given step is calculated, $v(t)$ from the most recent step, and $x(t)$ from the preceding step are used to describe the system at at time t. In addition, this implementation requires that $v(t-\frac{\delta t}{2})$ and $x(t)$ are somehow obtained for the beginning of a run. Therefore, runs usually start and end with $v(t - \frac{\delta t}{2})$ and $x(t)$.

The basic trotter algorithm breaks the contributions from short-range, F_{short}, and long-range, F_{long} forces. With $ntrott = \Delta t/\delta t$, the algorithm is:

$v \leftarrow v + F_{long}\frac{\Delta t}{2m}$
do $i = 1, ntrott$

$v \leftarrow v + F_{short}\dfrac{\delta t}{2m}$

$x \leftarrow x + v\,\delta t$

calculate F_{short}

$v \leftarrow v + F_{short}\dfrac{\delta t}{2m}$

end do

calculate F_{long}

$v \leftarrow v + F_{long}\dfrac{\Delta t}{2m}$

This implementation takes the system from $x(t) \rightarrow x(t + \Delta t)$ and $v(t) \rightarrow v(t + \Delta t)$. However, to integrate the algorithm into the structure of the cosmos code, it is necessary to arrange the step so that the force calculations take place at the end of a step rather than in the middle. In this description of the cosomos implementation, F_{long} is replaced by F_{pme}.

do $i = 1, ntrott$

clear F_{total}

if $(i = 1)$, then

calculate F_{pme}

$F_{total} = ntrott * pme$

endif

calculate F_{short}

$F_{total} = F_{total} + F_{short}$

$v \leftarrow v + F_{total}\dfrac{\delta t}{2m}$

perform velocity rattle

$v \leftarrow v + F_{total}\dfrac{\delta t}{2m}$

$x \leftarrow x + v\,\delta t$

perform position and velocity shake

end do

This arrangement breaks the algorithm into a part that sets up the the forces, and a part that propagates the system using velocity Verlet. The positions of the system

go from $x(t) \rightarrow x(t+\Delta t)$. However, the velocities between steps are not $v(t-\frac{\delta t}{2})$ because they are based on the forces $(ntrott\ F_{pme})+(F_{short})$ and do not correspond to an actual time. I will call the velocity at this time $v(\xi)$. After the first propagation step of the algorithm the velocities are moved to $v(t)$. Therefore it is easy to show how the velocities can be converted from $v(t-\frac{\delta t}{2})$, the form in which the velocities are stored, to $v(\xi)$, the form appropriate for the *trotter* algorithm. Here, F_{Verlet} equals the normal Verlet forces.

$$
\begin{aligned}
v(\xi) &= v(t) - (ntrott\ F_{pme}\frac{\delta t}{2m} + F_{short}\frac{\delta t}{2m}) \\
v(\xi) &= v(t) - F_{Verlet}\frac{\delta t}{2m} - (ntrott-1)\ F_{pme}\frac{\delta t}{2m} \\
v(\xi) &= v(t-\frac{\delta t}{2}) - (ntrott-1)\ F_{pme}\frac{\delta t}{2m}
\end{aligned}
$$

An additional startup routine is required to convert the velocities from $v(t-\frac{\delta t}{2}) \rightarrow v(\xi)$ by subtracting $[(ntrott-1)\ F_{pme}\frac{\delta t}{2m}]$ from $v(t)$. Also, a shutdown routine reverts the velocities to the standard from by adding $[(ntrott-1)\ F_{pme}\frac{\delta t}{2m}]$ to $v(\xi)$.

The Code. Few code changes are necessary to make the algorithm work with cosmos. One additional parameter, *ntrott* needs to be supplied to the program through the input file. The routine **mdfin** , which adjusts the velocities at the end of a run, is the only new routine in the program. All other changes were modifications to existing routines. Below is a brief description of their additional functions.

getopt reads the value for *ntrott* from the input file.

iter8 keeps track of the step number. For a step size Δt nrgfrc is passed the value of ntrott. For a step size δt nrgfrc is passed the number 1.

nrgfrc calls pme only if ntrott is greater than zero.

pme calculates pme forces. If *ntrott* equals zero, then the pme forces from the previous force call replace the total forces. If *ntrott* is greater than zero, then the pme forces are calculated, multiplied by ntrott, and added to the total forces.

main runs both mdstart and mdfin.

mdstart performs normal operations and, if *ntrott* is greater than one, subtracts $(ntrott-1)$ times the velocities from the pme, from all the velocities.

mdfin is called at the end of a run. If *ntrott* is greater then one, all velocities are adjusted by adding $(ntrott-1)$ times the velocities from the pme.

Method	δt	Δt	drift(wat-hex)	drift(protein)
Verlet	1	$\cdots$	-0.09	-0.2
	2	$\cdots$	-0.08	-1.18
	3	$\cdots$	-0.33	-3.34
	4	$\cdots$	-0.56	-6.04
	5	$\cdots$	-0.19	18.6
	6	$\cdots$	1.02	
	7	$\cdots$	18.6	
Trotter	1	2	-0.07	0.15
	1	3	-0.09	0.08
	1	4	-0.07	0.26
	1	5	-0.07	0.45
	1	6	-0.01	0.45
	1	7	-0.06	0.60
	1	8	0.05	1.21
	1	9	0.00	2.11
	1	10	1.44	9.36

Table 1: Comparison of energy drift per step in wat-hex and protein systems. The short-range time step, δt, and the long-range time step, Δt, are both fs. The energy drift, is given in units of $(kcal\, mol^{-1}\, ps^{-1})$.

Analysis

The performance of the cosmos-Trotter was tested with two systems: a water-hexane system(wat-hex)consisting of 6382 water molecules and 450 hexane molecules, and a water-protein system, (protein), consisting of 5020 water molecules and 5808 protein atoms. In all tests, the wat-hex system was run for 5 ps at 300K, and the protein system was run for 2 ps at 310K. The systems where tested with different different step sizes, Δt and δt. The total energy of the system, potential plus kinetic, is analyzed to see how stable it is throughout the run. A drift in energy indicates that the system is not conserving energy and implies that the trajectories can not be accurate. If the system is run a for a long time with an energy drift the temperature of the system starts to increase. This requires the velocities to be periodically sampled from a distribution to reset the temperature. This destroys the reversibility of the run.

Table 1 shows the energy drifts associated with the two systems. Figure 1. shows a comparison between the Verlet and Trotter methods for the wat-hex system. The Verlet can be expanded to a step size of about 2 fs but breaks down at longer time steps. The Trotter is stable up to 9 fs. In the protein tests, Fig. 2, the differences between the methods are more pronounced. Because the protein system contains more small molecular bonds which vibrate at a high frequency (on the order of

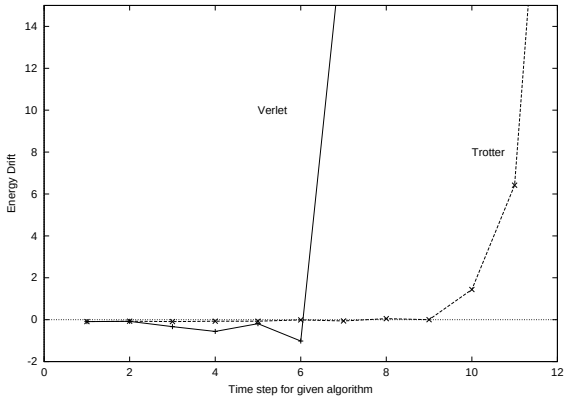

Figure 1: (wat-hex) Energy drift for step parameter. The step paramater is δt (fs) for Verlet, and Δt (fs) for Trotter. The energy drift is given in $\Delta E/ps$.

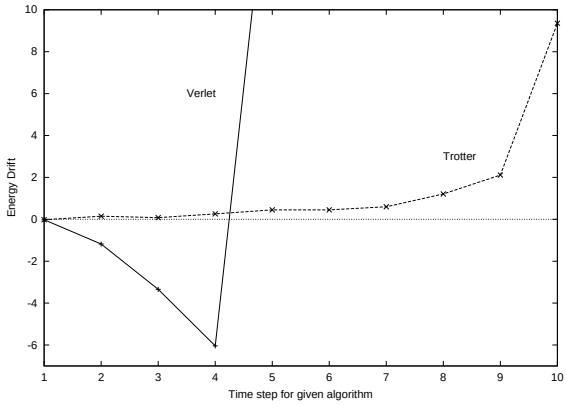

Figure 2: (protein) Energy drift for step parameter. The step paramater is δt (fs) for Verlet, and Δt (fs) for Trotter. The energy drift is given in $\Delta E/ps$

fs) the Verlet method breaks down immediately. The Trotter method continues to calculate the bond forces at 1 fs and remains quite stable for up to around $\Delta t = 7$ fs.

The Trotter method produced a increase in speed for both system. Fig. 3 shows the cpu time spent per step. These times are only approximations because that the program was not run with optimal speed settings and the conditions may vary. However, a decrease in time is shown. The wat-hex sytem goes from around 4.25 to around 2.5 seconds/step while maintaining stability. The protein system goes from around 5.9 to 4.2 seconds/step maintaining stability.

Additional Results

The Trotter algorithm is limited in part by the periodic oscillations that result from

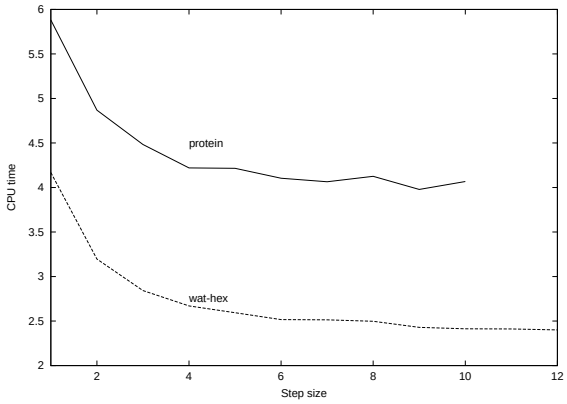

Figure 3: CPU time (seconds) per step. Note that the wat-hex system is stable to around $\Delta t = 9$ and the protein system is stable to around $\Delta t = 7$.

introducing "large" forces every Δt steps. If the longer time step is a multiple of the phase of the molecular bond vibrations it will start to drive vibrations and the simulation will fail.

One attempt to reduce this effect, an extrapolation/correction was tried on the system. [1] Rather than include $ntrott\,pme$ forces every *ntrott* steps, the pme forces were added every step, but only updated every *ntrott* steps. Since the system has a normal sized force each step, it was hoped that the outer time step could be increased without amplifying the bond occlusions. Additionally, a correction was put on the velocities every step to attempt to reduce the error introduced. This was a non-reversible system and the results were horrible. Even with very small step sizes the energy took off drastically.

An additional parameter that has not been tested is the cutoff length for the short-range forces. A cutoff length half the length of the simulation box has been suggested as a good setting[1] and adjusting the cutoff length for a particular system could improve the trotter results.

Furthermore, T. Schlick has suggested that stability can further be increased with the use of a position Verlet iterator instead of the velocity Verlet iterator.

Conclusions

This report has shown that is is possible to implement the Trotter algorithm into Cosmos with a decrease in simulation time. For both the water-hexane and the water-protein system, the PME forces could be calculated less frequently than the short range forces. This allows for longer, and more accurate simulations. When running Cosmos, care must be taken to record energies only from steps with full force calculations, and to ensure that a run ends on a full force calculation.

References

[1] E. Barth and T. Schlick. II. Extrapolation versus impulse in multiple-timestepping schemes: Linear analysis and applications to newtonian and langevin dynamics. *J. Chem. Phys.*, 109:1633–1642, 1998.

[2] B. Owenson, A. Pohorille, M.A. Wilson, M.H. New, and E. Darve. COSMOS — *A software package for COmputer Simulations of MOlecular Systems*. NASA — Ames Research Center, Moffett Field, CA 94035–1000, 1987.

[3] M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.*, 97(3):1990–2001, 1992.